UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Big Data Project: Social Network Analysis

Prof. Vincenzo Moscato, a.y. 2024-25

Students:
Saverio Dell'Aversana M63001613
Mauro Giliberti M63001676

Contents

Chapter 1

Introduction

Social media platforms serve as prolific generators of user-generated content, offering an invaluable resource for the analysis of user behavior, community dynamics, and information dissemination.
Traditional approaches to user analysis, such as text-based profiling and statistical behavior modeling, offer valuable insights but often fall short of providing a holistic, AI-driven interpretation of individual users. Text-based profiling, which includes techniques like sentiment analysis and topic modeling, captures thematic content but lacks personalized detail.
The emergence of Large Language Models (LLMs) has revolutionized text-based tasks, demonstrating remarkable capabilities in generating human-like text and excelling in complex reasoning and communication tasks.
This project addresses the challenge of enabling LLMs to emulate specific social media users by structuring and extracting key behavioral insights from their online activity. **Named Entity Recognition (NER)** and **Entity Linking** are employed to identify and categorize key entities within user-generated content, providing a structured understanding of the subjects users discuss. **Topic Extraction** helps to discern the thematic preferences and recurring subjects in a user's communication. Furthermore, **Retrieval-Augmented Generation (RAG)** allows the LLM to contextualize and attribute social media content based on user-specific characteristics, behavioral trends, and thematic affinities.

Chapter 2

Methodology Pipeline

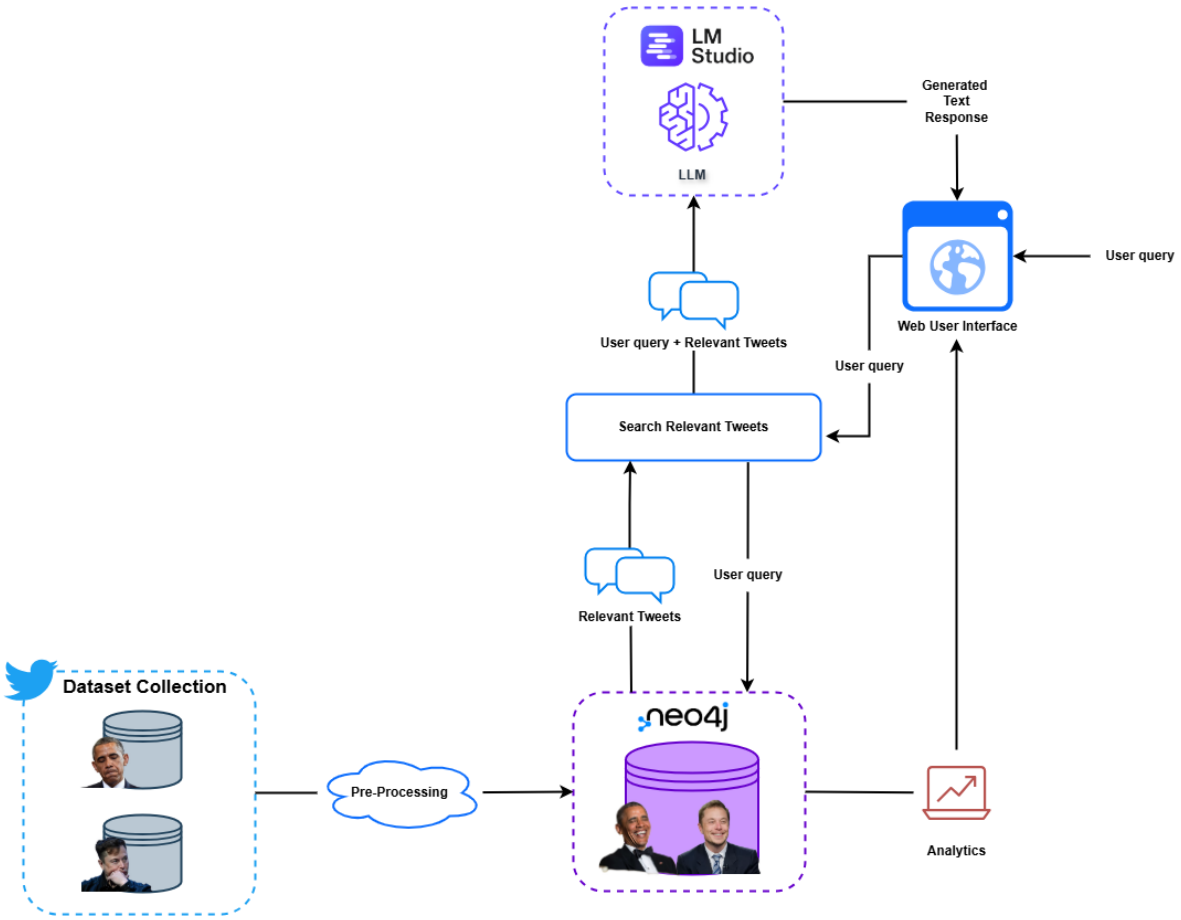

Figure 2.1: General Pipeline

This chapter outlines the overall methodology pipeline for the project, as depicted in Figure 2.1. The system is designed to leverage social media data for user characterization and to enable an LLM to interpret and generate user-specific content. The pipeline consists of several key stages, from data collection and preprocessing to LLM interaction and analytical output.

2.1 Dataset Management and User Characterization

The initial phase of the project involves the collection of social media data, tweets, which forms the basis for understanding user behavior and content generation patterns. This raw data undergoes a necessary **pre-processing stage** to ensure its suitability for subsequent analysis. The cleaned and structured data is then **ingested into a Neo4j graph database**. This choice is strategic, as Neo4j excels at modeling complex relationships inherent in social media, facilitating the characterization of users through their interactions, content, and thematic connections. This graph-based storage acts as a central repository for structured user profiles and social media content.

2.2 Interactive LLM Workflow

The core of the system's interaction capabilities is managed through a **Web User Interface**. This interface serves a dual purpose: it allows users to initiate queries and visualize analytics. When a user submits a query via the interface, this action triggers a process to retrieve relevant tweets. These relevant tweets are dynamically searched within the Neo4j database, ensuring that the retrieved content is contextually aligned with the user's query and potentially with specific user profiles.
The user query, along with these relevant tweets, is then fed into the **Large**

Language Model (LLM). The diagram illustrates the use of **LM Studio** as a tool for running the LLM. The LLM processes these combined inputs to generate a coherent and contextually appropriate text response. This generated response is then sent back to the Web User Interface, where it is displayed to the user, completing the interactive cycle. This iterative process allows the LLM to provide insights and generated content that are informed by real social media data and user context.

2.3 Analytics

Beyond the interactive query system, the pipeline also incorporates an **Analytics component**. This module leverages the rich, interconnected data stored in the Neo4j database to extract valuable insights. These analytics can cover various aspects of social media behavior, such as identifying trending topics, analyzing community affiliations, and comparing user activity patterns. This component provides a deeper understanding of individual and collective social media dynamics, complementing the LLM's interpretive abilities.

Chapter 3

Dataset and Preprocessing

3.1 Datasets

For the purpose of this project, we focused on analyzing social media data using two publicly available Twitter datasets. The first dataset contains tweets from Elon Musk, spanning from 2010 to 2017. The second dataset includes tweets posted by Barack Obama from 2012 to 2019.

3.1.1 Dataset Selection and Features

To ensure a consistent and focused analysis, we retained only a subset of features from each dataset:

- **Timestamp** The date and time when the tweet was posted.
- **Tweet Text** The full textual content of each tweet.
- **Number of Likes** The number of likes received by the tweet.
- **Number of Retweets** The number of retweets, reflecting the tweet's engagement.

These fields provided the necessary basis for temporal, semantic, and popularity based analysis across both datasets.

3.2 Preprocessing and Entity Based Topic Identification

The preprocessing pipeline was designed to extract meaningful information from raw tweet text and prepare it for downstream tasks such as classification and sentiment analysis. The key preprocessing steps included:

- **Named Entity Recognition**: A NER model has been applied to identify named entities within the tweets, such as persons, organizations, locations, and products. This step enabled to extract the core semantic components from the tweet content.

- **Topic Derivation from Entities**: The extracted named entities served as the basis for classifying tweet topics. These topics were not predefined but were inferred dynamically, depending on the context and frequency of entities across the dataset, which made this stage, and consequently the entire pipeline, semi-automatic. To enhance the reliability of topic classification, a **confidence thresholding strategy** was employed. Specifically, for each topic, the median confidence score across all associated tweets was calculated. Tweets with a predicted topic confidence score below this median value were excluded from further processing. In this analysis, an upper threshold of 0.6 was set, ensuring that also if the median value was pretty high, significant tweets were not excluded. This per-topic thresholding approach allowed for adaptive filtering based on the distribution of confidence scores within each topic, thereby improving the overall quality of the topic assignments.

- **Topic-Based and Entity-Based Sentiment Analysis** Following the identification of topics and entities, we performed a sentiment analysis for each tweet, leveraging both topics and entities. This dual-layer sentiment analysis aimed to better understand the stance,

tone, and emotional valence associated with the various subjects and actors discussed in the tweets.

Chapter 4

Database choice and Data Ingestion

4.1 Introduction to Graph Databases

Among various database paradigms, graph databases have emerged as a powerful solution for managing interconnected data. Unlike traditional relational databases that store data in tables with predefined schemas, or NoSQL databases that handle unstructured data, **graph databases** represent data as **nodes** (***entities***) and **edges** (***relationships***) between them. This structure inherently reflects the relationships between data points, making them particularly adept at handling highly connected datasets.
Key characteristics of graph databases include:

- **Nodes**: Represent entities. Each node can have properties, which are key-value pairs describing the node.
- **Relationships (Edges)**: Connect nodes and represent how entities are associated. Relationships can also have properties, such as timestamps or weights.
- **Property Graph Model**: Most modern graph databases adhere to this model, allowing both nodes and relationships to have properties.

- **Schema Flexibility**: Graph databases are generally schema-flexible, allowing for easier evolution of the data model as requirements change.
- **Traversal Optimization**: They are highly optimized for traversing relationships, making complex queries involving multiple hops significantly faster than in relational databases.

The ability to directly model and query relationships makes graph databases ideal for domains such as social networks, recommendation systems, fraud detection, and knowledge graphs, where understanding connections is paramount.

4.2 Neo4j: The Chosen Graph Database

For this project, **Neo4j** was selected as the graph database of choice. Neo4j is a leading native graph database, meaning it stores and processes data in its native graph structure, providing exceptional performance for graph traversals and complex queries.
The advantages that led us to the selection of Neo4j include:

- **Native Graph Processing**: Its architecture is specifically designed for graph data, offering superior performance for queries that navigate relationships, which is crucial for analyzing social media interactions.
- **Cypher Query Language**: Neo4j uses Cypher, a declarative, SQL-like graph query language that is intuitive and highly expressive for querying and manipulating graph data.
- **Scalability**: Neo4j offers various scaling options, allowing it to handle growing datasets and increasing query loads.
- **Data Modeling Flexibility**: Its flexible schema allows for iterative development and adaptation of the data model without rigid constraints, which is beneficial given the evolving nature of social media data.

These benefits make Neo4j an excellent fit for modeling the intricate web of interactions, content, and entities found within social media, enabling efficient storage and retrieval of complex user-related information.

4.3 Data Ingestion into Neo4j

Once the social media data was collected and pre-processed to ensure cleanliness and consistency, it was then ingested into the Neo4j database. The ***ingestion process*** involved creating nodes and relationships based on the parsed data, effectively transforming raw text and metadata into a connected graph structure. This was primarily achieved using Cypher, Neo4j's powerful graph query language.
The data ingestion was performed in two main parts.

4.3.1 Creation of Tweet Nodes

The first step focused on creating nodes representing individual tweets. Each row from the pre-processed dataset (assumed to be in a CSV format) was loaded, and a **`Tweet`** node was created or merged based on unique identifiers like date and text. Properties such as `retweets`, `likes`, `topic`, `confidence`, `sentiment`, `sentiment_confidence`, and `author` were assigned to each `Tweet` node.

```
LOAD CSV WITH HEADERS FROM 'file:///dataset.csv' AS
    row
MERGE (t:Tweet {
  date: row.Date,
  text: row.`Text`
})
SET t.retweets = toInteger(row.Retweets),
    t.likes = toInteger(row.Likes),
    t.topic = row.topic,
```

```
    t.confidence = toFloat(row.confidence),
    t.sentiment = row.sentiment,
    t.sentiment_confidence = toFloat(row.
      sentiment_confidence),
    t.author = row.Author;
```

Listing 4.1: Tweet node creation

4.3.2 Creation of Entity Nodes and MENTIONS Relationships

The second part of the ingestion process involved identifying and creating **`Entity`** nodes and establishing **`MENTIONS`** relationships between `Tweet` nodes and these `Entity` nodes. The `entities` column in the dataset, which contained a list of identified entities and their types, was processed to extract individual entities. For each unique entity, an `Entity` node with `name` and `type` properties was created or merged. Subsequently, a `MENTIONS` relationship was established from the `Tweet` node to the corresponding Entity node. This step is critical for linking tweets to the specific subjects, organizations, or people they discuss.

```
LOAD CSV WITH HEADERS FROM 'file:///dataset.csv' AS
    row

WITH row, replace(replace(replace(replace(replace(
  replace(row.entities, "[[", ""), "]]", ""), "[",
  ""), "]", ""), "'", ""), "\"", "") AS cleaned

WITH row, split(cleaned, ", ") AS parts

WITH row, [i IN range(0, size(parts)-1)
          WHERE i % 2 = 0 AND i + 1 < size(parts)
```

```
            | {name: trim(parts[i]), type: trim(parts
              [i+1])}] AS entityTuples

UNWIND entityTuples AS entity

WITH row, entity
WHERE entity.name IS NOT NULL AND entity.type IS
  NOT NULL AND entity.name <> "" AND entity.type <>
   ""

MERGE (e:Entity {name: entity.name, type: entity.
  type})

WITH row, e
MATCH (t:Tweet {date: row.Date, text: row.`Text`})
MERGE (t)-[:MENTIONS]->(e);
```

Listing 4.2: Entity node and MENTIONS relationships creation

4.4 Graph Model

This data model results in the creation of two primary types of nodes:

- **Tweet Nodes**: Represent individual social media posts, containing metadata such as date, text, likes, retweets, sentiment, and author.
- **Entity Nodes**: Represent distinct named entities (e.g., persons, organizations, locations) extracted from the tweet text, along with their types.

The fundamental relationship connecting these nodes is **MENTIONS**. This graph representation is incredibly powerful for several reasons, especially when dealing with complex social media data:

- **Intuitive Data Representation**: It naturally maps the way social media content and its subjects are interconnected, making the data model easy to understand and visualize.
- **Efficient Data Retrieval**: Complex queries, such as finding all tweets mentioning a specific entity, or identifying entities frequently mentioned together, become highly efficient due to optimized graph traversals.
- **Foundation for Advanced Analytics**: This interconnected graph serves as a robust foundation for more advanced analytics, such as community detection, influence analysis, and the development of rich knowledge graphs that can be leveraged by LLMs for more nuanced understanding and generation of user-specific content.

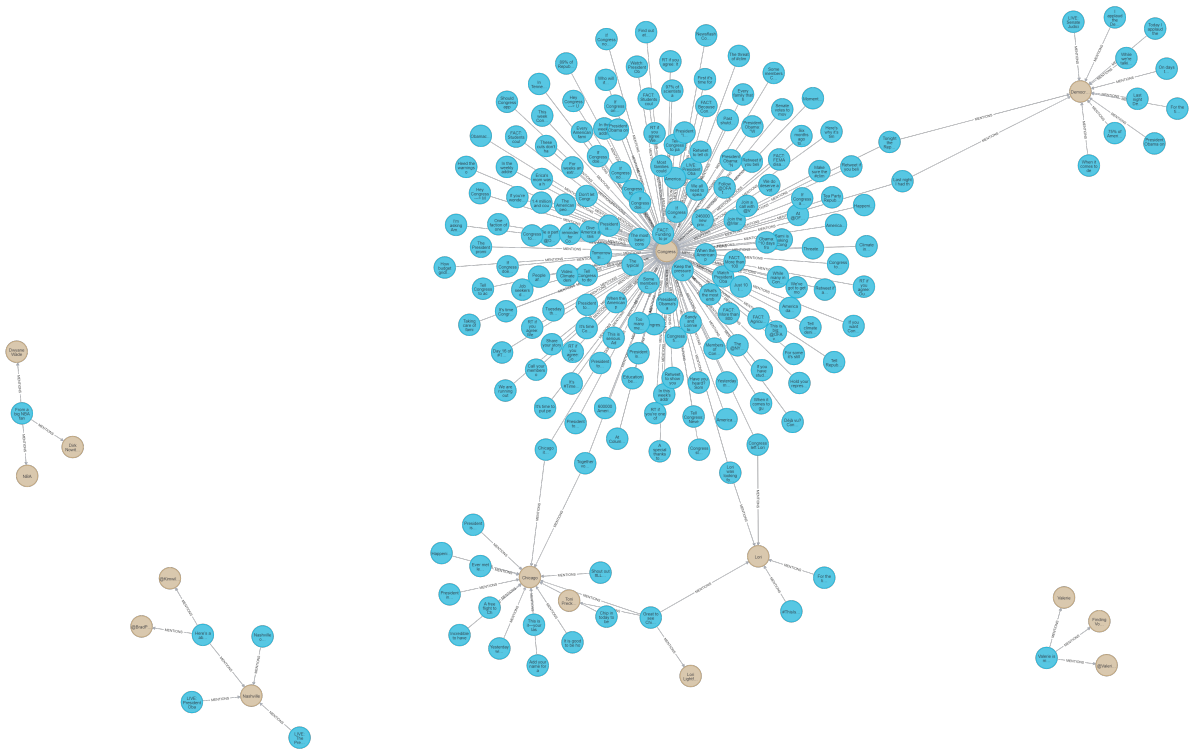

Figure 4.1: A partial view of Neo4j graph

Chapter 5

Tweet Generation and Tweet Identification with RAG

5.1 Retrieval-Augmented Generation (RAG)

To enhance the performance and factual grounding of the LLM, this project heavily leverages the **Retrieval-Augmented Generation (RAG)** technique. RAG combines the strengths of large pre-trained language models with the ability to retrieve relevant information from an external knowledge base. Instead of relying solely on the knowledge encoded during its training, the LLM first retrieves pertinent documents or data snippets from a specified data source. This retrieved information then serves as additional context for the LLM during its generation process.
The key advantages of employing RAG are:

- **Improved Factual Accuracy**: By grounding responses in retrieved data, RAG significantly reduces the likelihood of the LLM *hallucinating* information.
- **Privacy**: Companies can avoid feeding LLMs with sensitive data

whose spread could be dangerous.

- **Up-to-Date Information**: The external knowledge base can be continuously updated, allowing the LLM to access the most current information without requiring retraining.
- **Explainability**: Responses are often more transparent, as the generated output can be traced back to the specific retrieved documents that influenced it.

In this project, the external knowledge base is the social media dataset stored and managed within the Neo4j graph database, providing a rich, interconnected source of user-specific tweets.

5.1.1 Chosen Large Language Model

The **`meta-llama-3.1-8b-instruct`** model was utilized, managed and served locally via `LM Studio`. `meta-llama-3.1-8b-instruct` is an 8-billion parameter version of Meta's Llama 3.1 family of models, fine-tuned for instruction following. It is a powerful **transformer-based** architecture, which is the foundational design for most state-of-the-art LLMs. Transformers are characterized by their multi-head self-attention mechanisms, allowing them to weigh the importance of different parts of the input sequence when processing each word. The `context lenght` is set to 8192 to let the model capture long-range dependencies and complex linguistic patterns effectively.

5.2 LLM Applications within the Pipeline

The LLM is leveraged for two distinct yet complementary purposes within the project: ***Author Identification*** and ***Tweet Generation in Author's Style***. Both applications benefit from the RAG approach, using the Neo4j database to provide relevant context.

5.2.1 Author Identification

The author identification task aims to predict the most likely author (Obama or Musk, or neither) of a given unseen tweet. This process involves several steps orchestrated by the FastAPI backend:

1. **Topic Classification**: Upon receiving a new tweet, the first step is to classify its dominant topic in the same way seen during preprocessing. This helps narrow down the search space for relevant contextual tweets.

2. **Context Retrieval from Neo4j**: Based on the identified topic, the system queries the Neo4j database to retrieve a collection of existing tweets that share the same topic, regardless of author.

3. **Embedding and Similarity Search**: The retrieved `corpus` tweets and the new `query` tweet are then transformed into numerical vector representations (**embeddings**) using a `SentenceTransformer` model, specifically **`all-MiniLM-L6-v2`**. These embeddings capture the semantic meaning of the tweets. A **`FAISS`** (**Facebook AI Similarity Search**) index is then used to efficiently search for the top 10 most semantically similar tweets from the corpus to the query tweet. This step ensures that the most relevant contextual examples are selected.

4. **Prompt Construction**: The selected context tweets, along with their respective authors, are formatted into a concise string. This context, combined with the new tweet and a direct question regarding authorship, forms the final prompt that is sent to the LLM. An example prompt structure is shown in Figure 5.1.

5. **LLM Inference and Streaming**: The LLM processes this prompt to determine the likely author and provide an explanation. The response is streamed back to the frontend allowing for a real-time user

experience as the LLM generates its output. The `temperature` is set to 0.1 for more deterministic responses in this task.

5.2.2 Tweet Generation in Author's Style

The second application allows users to request a new tweet generated in the style of a specific author on a given topic.

1. **Context Retrieval from Neo4j**: Given a target author and topic, the system retrieves a sample of up to 20 existing tweets by that specific author on that particular topic from the Neo4j database.

2. **Prompt Construction**: This includes the sampled context tweets and explicitly requests the LLM to generate a new concise and engaging tweet in the style of the specified author about the given topic, with instructions to avoid extra explanations Figure 5.1.

3. **LLM Inference and Streaming**: The LLM processes these prompts and generates the new tweet. Similar to author identification, the response is streamed for a dynamic user experience. The `temperature` for generation is set to 0.7 to allow for more creativity and variation in the output, while remaining consistent with the learned style.

Both LLM applications highlight the power of combining a structured knowledge base (Neo4j) with a sophisticated generative model (meta-llama-3.1-8b-instruct) via the RAG paradigm, enabling the system to perform nuanced tasks like author attribution and style-transfer generation.

Prompt Author Identification

System: You are an expert in tweet author attribution. Provide concise and accurate explanations based on the context.

User: I will provide you with a list of tweets.
Tweets: {context_str}
Now, consider this new tweet {data.tweet}
Question: Could this tweet have been written by Obama, Musk or neither?
Answer and give a brief explanation.

Prompt Tweet Generation

System: You are an expert in generating tweets in the style of a specific author. Your task is to produce a tweet that closely mimics the writing style, tone, and common vocabulary of {author} on the given topic.

User: Based on these example tweets by {author} on the topic of {topic}:
{context_tweets}
Now, generate a new tweet in the style of {author} about {topic}.
The tweet should be concise and engaging.
Do NOT include any explanations or extra text, just the tweet itself.
Answer and give a brief explanation.

Figure 5.1: LLM prompts: on the left the Author Identification Prompt; on the right the Tweet Generation Prompt

Chapter 6

Analytics

The **analytics** provide valuable insights into individual and collective social media behaviors, enhancing the overall understanding derived from the dataset.

- **Likes by Year for Topic and Author:** This analytic retrieves the total number of likes for tweets on a specific topic, filtered by a particular author, grouped by year.

  Purpose: To understand the popularity trend of a topic for a given author over the years.

- **Topic Trend by Month and Year:** This analytic determines the prevalence of different topics within tweets for a specified year and author (or all authors), broken down by month.

  Purpose: To identify monthly topic trends and their evolution within a given year.

- **Top Tweets by Metric:** This analytic fetches the top tweets based on a chosen metric (likes or retweets), with an optional filter for a specific author.

  Purpose: To highlight the most engaging tweets by a particular author or across all authors.

- **Average Sentiment by Topic:** This analytic calculates the weighted average sentiment for each topic across all tweets, considering the sentiment confidence.

  Purpose: To gauge the general emotional tone associated with different topics.

- **Average Sentiment per Year by Author:** This analytic computes the weighted average sentiment of an author's tweets for each year.

  Purpose: To observe how an author's overall sentiment has changed over time.